Computational Linguistics 2014-2015

- **Walter Daelemans** (walter.daelemans@uantwerpen.be)
- **Guy De Pauw** (guy.depauw@uantwerpen.be)
- **Mike Kestemont** (mike.kestemont@uantwerpen.be)

http://www.clips.uantwerpen.be/cl1415

Practical

Location	P0.11 (Scribanihuis)
Reading material	• D. Jurafsky & J.H. Martin (2009) Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2nd ed). Pearson Education, USA. • Natural Language Processing with Python
Software	Python 3.4 and NLTK: Installation Instructions
Evaluation	Take-home assignments and oral examination
Lecturers	Walter Daelemans: walter.daelemans@uantwerpen.be Mike Kestemont: mike.kestemont@uantwerpen.be Guy De Pauw: guy.depauw@uantwerpen.be

Program

Session	Day	Date	Chapter	Topic	Reading Assignment	Slides	Take-home Assignment
1	Monday	29/9/2014	**Python**	Session 1 - Variables	See Github		
2	Thursday	2/10/2014	**Python**	Session 2 - Collections			
3	Monday	6/10/2014	**Python**	Session 3 - Conditions (and an introduction to loops)			
4	Thursday	9/10/2014	**Python**	Session 4 - Loops			
5	Monday	13/10/2014	**Python**	Session 5 - Reading and writing to files			
6	Thursday	16/10/2014	**Python**	Session 6 - Writing your own Functions and importing packages			
7	Monday	20/10/2014	**Python**	Session 7 - Regular Expressions in Python			
8	Thursday	23/10/2014	**Python**	Session 8 - Advanced looping in Python and list comprehensions			
9	Monday	27/10/2014	**Theory**	Introduction to Computational Linguistics	Jurafsky & Martin: Chapter 1		
10	Monday	3/11/2014	**Theory**	Regular Expressions and Finite State Automata & Transducers	Jurafsky & Martin: Chapter 2; Chapter 3		
	Monday	10/11/2014	**Remembrance day: no session**				
11	Monday	17/11/2014	**Theory**	Part-of-Speech Tagging	Jurafsky & Martin: Chapter 5 (not 5.5, 5.8 and 5.9)		
12	Monday	24/11/2014	**Theory**	Syntactic Analysis & Parsing	Jurafsky & Martin: Chapter 12 (not 12.7.2, 12.8); Chapter 13 (not 13.4.1, 13.4.2, 13.5.1)		
13	Monday	1/12/2014	**Theory**	Probabilistic Methods	Jurafsky & Martin: Chapter 4.1, 4.2 and 4.3; Chapter 5.5 and 5.9; Chapter 14.1, 14.3 and 14.4		
14	Monday	8/12/2014	**Theory**	Word Sense Disambiguation	Jurafsky & Martin: Chapter 19.1, 19.2, 19.3, Chapter 20 (20.1->20.5)		
15	Monday	15/12/2014	**Theory**	Sentence semantics and discourse; Information extraction	Jurafsky & Martin: Chapter 21; Chapter 22		

Chapter 5

Morpho-Syntactic Part-of-Speech Tagging

Part-of-Speech Tagging

Assigning morpho-syntactic categories (part-of-speech tags, parts of speech, pos tags) to words in a sentence:

Morpho-Syntactic Categories:

- **CLOSED CLASS**
  - determiners: the, a
  - prepositions: in, out, over,…
  - auxiliary verbs: can, must, should, would,…
  - numbers: one, two, three,…
  - pronouns: I, you, we, he, …
  - conjunctions: and, but, or, as, if, when
- **OPEN CLASS**
  - nouns: cat, dog, paper, computer,… **also** proper nouns
  - verbs: work, cry, fly,… **but not** auxiliary verbs, modals
  - adjectives: green, blue, nice,…
  - adverbs: nicely, home, slowly, …

Part-of-Speech Tagging

- Dionysius Thrax of Alexandria (100BC): 8 POS tags
- High School: 8 POS tags
- Penn Treebank: 45 POS tags
- Brown Corpus: 87 POS tags
- C7 tagset: 146 POS tags

Penn Treebank Tag Set

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person sg present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Part-of-Speech Tagging

Why is part-of-speech tagging useful?

- Text-to-Speech
  e.g. **con**tent (noun) vs con**t**ent (adjective)

- Information Retrieval:
  e.g. terrorist **bombing**: noun
  also look for 'bombing+s'

- Generally considered as first step in Syntactic Disambiguation

- The seminal annotation task in NLP

Part-of-Speech Tagging

First step in Syntactic Analysis:

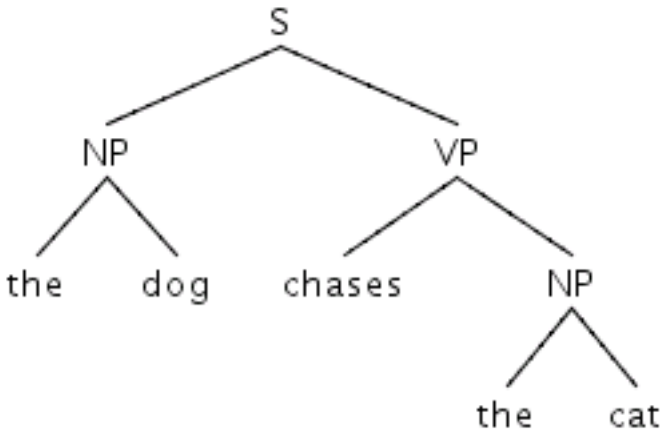
Grammar:

$S \rightarrow NP\ VP$

$NP \rightarrow$ the dog

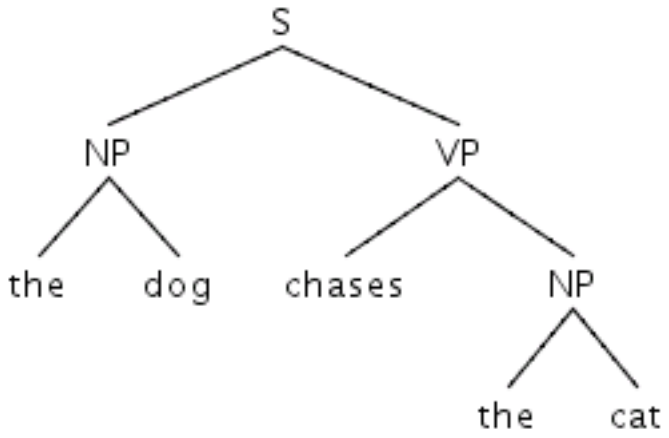
$NP \rightarrow$ the cat

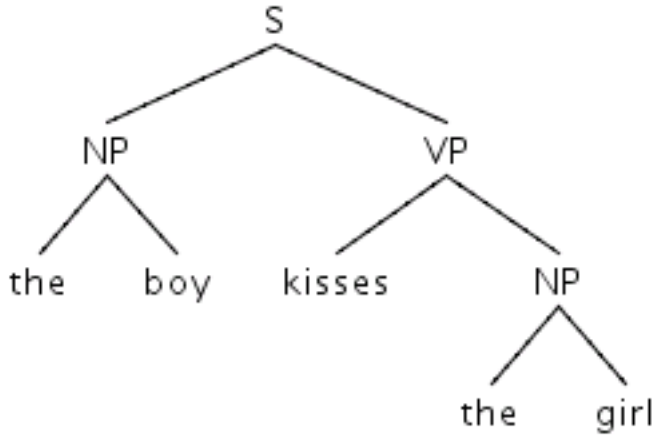
$VP \rightarrow$ chases NP



Part-of-Speech Tagging

Extend Grammar to cover two structures



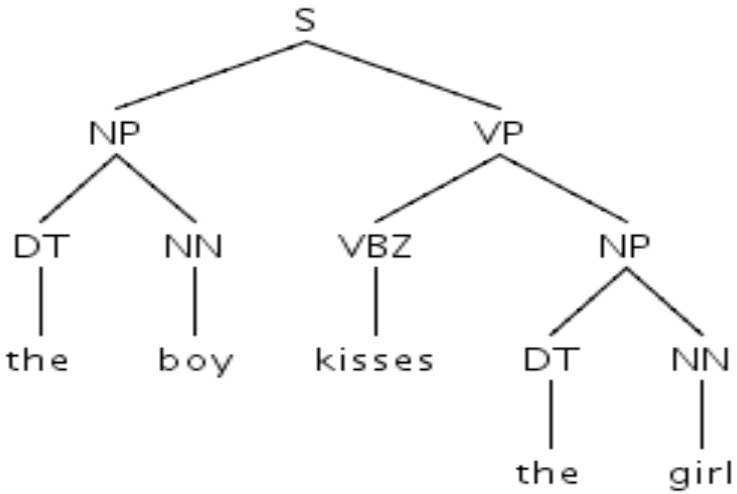


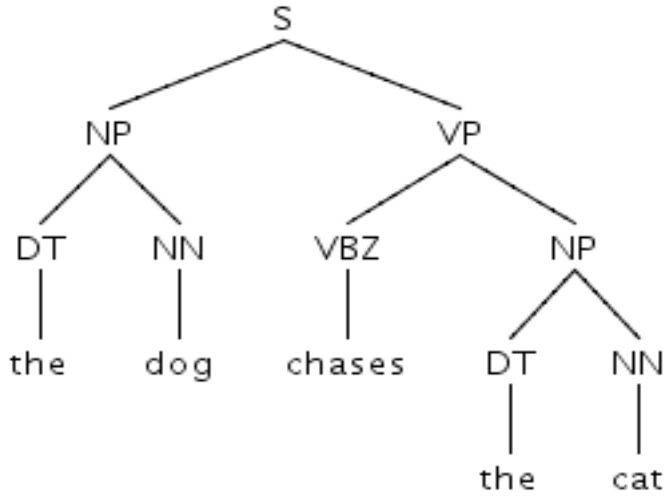
Grammar:

S → NP VP
NP → the dog
NP → the cat
NP → the boy
NP → the girl
VP → chases NP
VP → kisses NP

Part-of-Speech Tagging

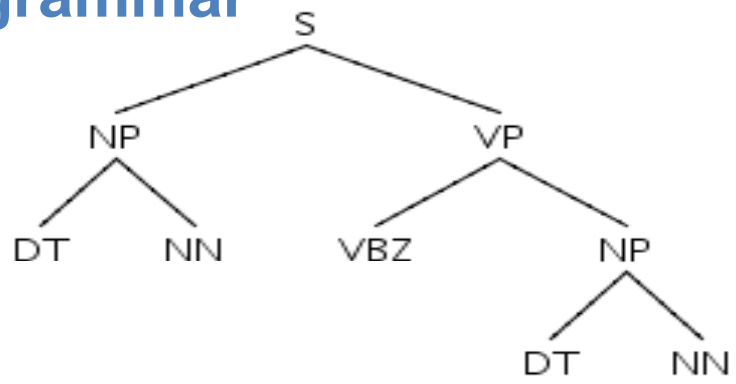
Use Part-of-Speech Tags to prevent explosion of grammar

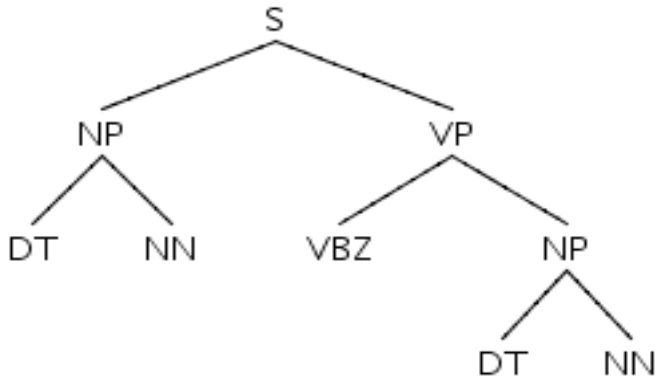




Part-of-Speech Tagging

Use Part-of-Speech Tags to prevent explosion of grammar

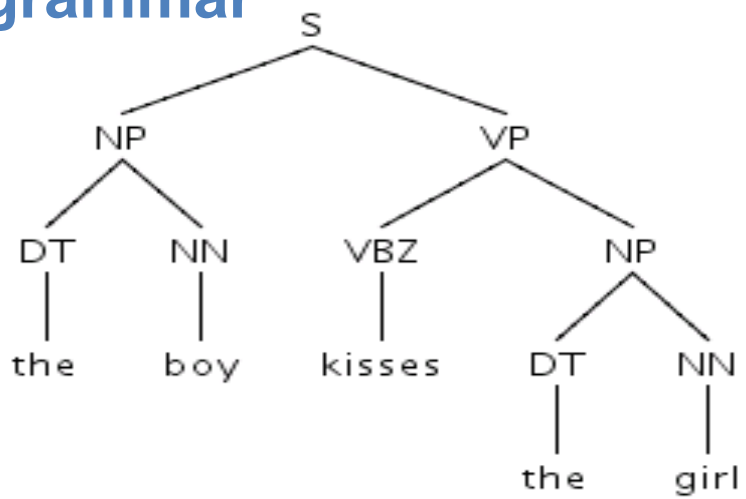


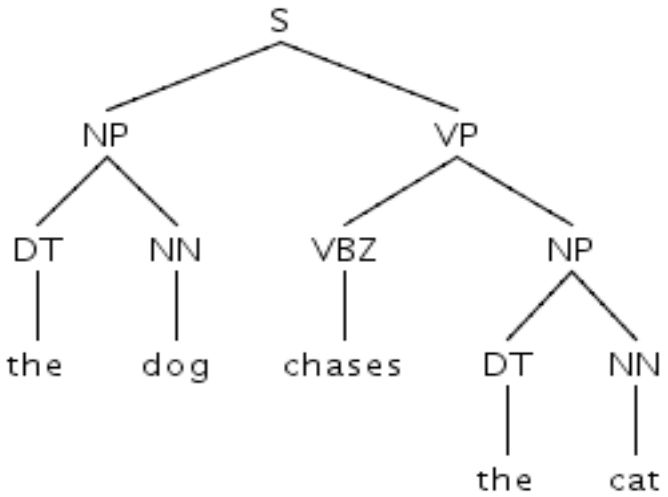


Grammar: **S → NP VP**
NP → DT NN
VP → VBZ NP

Part-of-Speech Tagging

Use Part-of-Speech Tags to prevent explosion of grammar

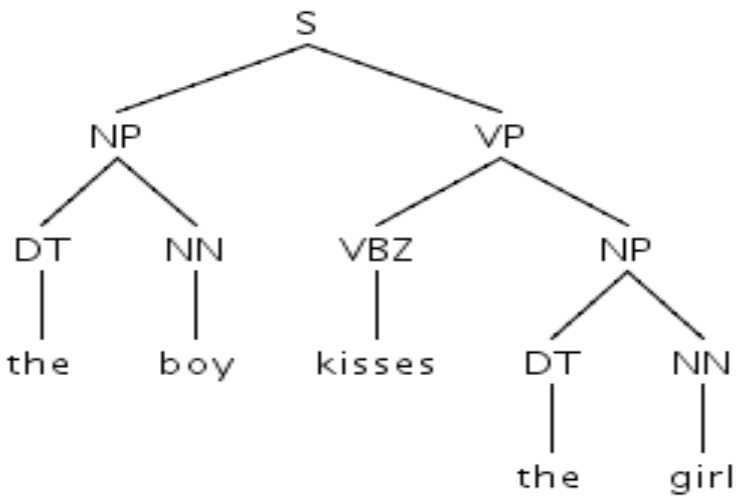


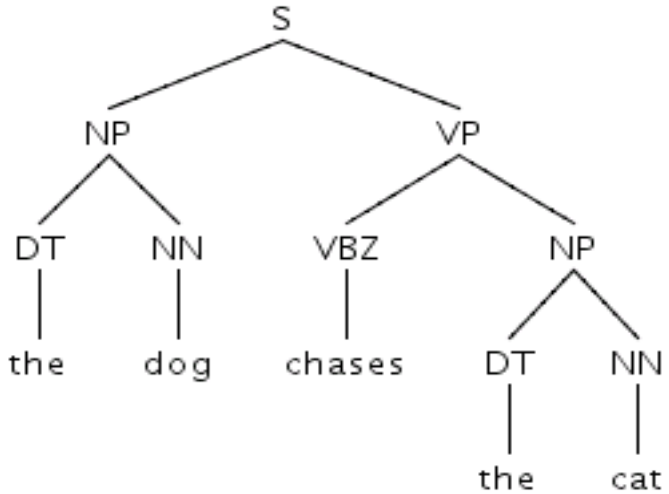


Grammar: S → NP VP
NP → DT NN
VP → VBZ NP

Lexicon: DT → the
NN → cat, dog, boy, girl
VBZ → kisses, chases

Part-of-Speech Tagging





- Part-of-Speech Tagging introduces new level to tree structure
- Unary Relation
- Why is this difficult?

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	adverb	noun	verb	verb
	noun	article	verb		adjective
	verb				adverb

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	adverb	noun	verb	verb
	noun	article	verb		adjective
	verb				adverb

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	adverb	noun	verb	verb
	noun	article	verb		adjective
	verb				adverb

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	adverb	noun	verb	verb
	noun	article	verb		adjective
	verb				adverb

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	adverb	noun	verb	verb
	noun	article	verb		adjective
	verb				adverb

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	adverb	noun	verb	verb
	noun	article	verb		adjective
	verb				adverb

Ambiguity in POS tagging

e.g.	Can	this	tag	be	better
	modal	article	noun	verb	adjective

Part-of-Speech Tagging is a typical NLP problem:

::::disambiguation in context::::

- 1 item with different possible categories (cf. word-sense disambiguation)
- Find correct category through:
  - **CONTEXTUAL CLUES**

    e.g. previous word is a determiner
  - **MORPHOLOGICAL CLUES**

    e.g. word ends in -er

Methods for POS Tagging

Manually Constructed

- rule-based methods
- based on insights from theoretical linguistics

- *Garside et al (1987)*
- *Klein & Simmons (1963)*
- *Green& Rubin (1971)*
- *Karlsson (1995)*
- *Voutilainen (1995)*
- *Oflazer-Kuruoz (1994)*
- *Chanod & Tapanainen (1995)*

Data-Driven/Inductive Taggers

- Probabilistic Methods
- Machine Learning Methods
- faster development, better results

- *Cardie (1994-1996):* ***Case-Based***
- *Daelemans (1996):* ***MBT*** *(**MBL**)*
- *Schmid (1994):* ***Decision Tree***
- *Nakumara (1980):* ***Neural Networks***
- *Cutting (1992):* ***HMM***
- *Ratnaparkhi (1996):* ***MXPOST*** *(**Maximum Entropy**)*
- *Thorsten Brants (2002):* ***TnT*** *(**statistical**)*

Brill 1992: ***Transformation-based Part-of-Speech Tagging***

Rule-Based Tagging

vb ENGTWOL (1995)

2 levels: 1. Lexicon-lookup

find POS-tag candidates for a word

2. Handcrafted disambiguation rules (3744)

single out one POS-tag

Rule-Based Tagging

Level 1: Lexicon-lookup

Pavlov	NNP(NOM SG)
had	VBN (SVO) VBD (SVO)
shown	VBN (SVOO/SVO/SV)
that	RB PRP(DEM SG) DT WDT
salivation	NN(NOM SG)
…	

Level 2: Rules / Constraints

Given input "that"

if	(+1 JJ/RB); (+2 SENT-LIM); (-1 NOT SVOC/A)	*Is it really that bad?* " ↔ *Do you consider that odd?*
then	delete all non-RB tags	
else	delete RB-tag	

Rule-Based Tagging

Level 1: Lexicon-lookup

Pavlov	**NNP(NOM SG)**
had	VBN (SVO) VBD (SVO)
shown	VBN (SVOO/SVO/SV)
that	RB PRP(DEM SG) DT WDT
salivation	NN(NOM SG)
…	

Level 2: Rules / Constraints

Given input any_word

if (/^[A-Z][a-z]+/);
(-1 NOT SENT-LIM);

then assign NNP tag

else nothing

Methods for POS Tagging

Manually Constructed

- rule-based methods
- based on insights from theoretical linguistics

- *Garside et al (1987)*
- *Klein & Simmons (1963)*
- *Green& Rubin (1971)*
- *Karlsson (1995)*
- *Voutilainen (1995)*
- *Oflazer-Kuruoz (1994)*
- *Chanod & Tapanainen (1995)*

Data-Driven/Inductive Taggers

- Probabilistic Methods
- Machine Learning Methods
- faster development, better results

- *Cardie (1994-1996):* ***Case-Based***
- *Daelemans (1996):* ***MBT*** *(**MBL**)*
- *Schmid (1994):* ***Decision Tree***
- *Nakumara (1980):* ***Neural Networks***
- *Cutting (1992):* ***HMM***
- *Ratnaparkhi (1996):* ***MXPOST*** *(**Maximum Entropy**)*
- *Thorsten Brants (2002):* ***TnT*** *(**statistical**)*

Brill 1992: ***Transformation-based Part-of-Speech Tagging***

Data-Driven POS tagging

- From mid 90s: established data-driven methods for POS tagging of Indo-European languages
  - Many publically available tools: Brill, MBT, MXPOST, TnT, SVMTool, CRF++, TreeTagger, CLAWS, QTAG, Xerox, ...
    - WSJ corpus (English): ±97%

      http://www.clips.ua.ac.be/cgi-bin/webdemo/MBSP-instant-webdemo.cgi
    - French Treebank (French): ±97%
    - CGN corpus (Dutch): ±97%

      http://ilk.uvt.nl/cgntagger/
    - Negra corpus (German): ±97%
    - MULTEXT-East (Slovene): ±90%
    - Helsinki Corpus of Swahili: ±98%

      http://aflat.org/node/10
    - Northern Sotho: ±94%

      http://aflat.org/node/177

Needed: annotated corpus

The DT
cafeteria NN
remains VBZ
closed JJ
PERIOD PERIOD
<utt>
Some DT
analysts NNS
argued VBD
that IN
there EX
wo MD
nSQt RB
be VB
a DT
flurry NN
of IN
takeovers NNS
because IN
the DT
industry NN
SQs POS
continuing JJ
capacity-expansion JJ
program NN
is VBZ
eating VBG
up RP
available JJ
cash NN
PERIOD PERIOD
<utt>

Probabilistic POS Tagging

- Requires annotated corpus

  can/MD the/DT tag/NN be/VB better/NN
- Unigram: P(tag|word)

  frequency of the tag for this word in corpus
- More on probabilistic POS tagging on 18/11

Methods for POS Tagging

Manually Constructed

- rule-based methods
- based on insights from theoretical linguistics

- *Garside et al (1987)*
- *Klein & Simmons (1963)*
- *Green& Rubin (1971)*
- *Karlsson (1995)*
- *Voutilainen (1995)*
- *Oflazer-Kuruoz (1994)*
- *Chanod & Tapanainen (1995)*

Data-Driven/Inductive Taggers

- Probabilistic Methods
- Machine Learning Methods
- faster development, better results

- *Cardie (1994-1996):* ***Case-Based***
- *Daelemans (1996):* ***MBT*** *(**MBL**)*
- *Schmid (1994):* ***Decision Tree***
- *Nakumara (1980):* ***Neural Networks***
- *Cutting (1992):* ***HMM***
- *Ratnaparkhi (1996):* ***MXPOST*** *(**Maximum Entropy**)*
- *Thorsten Brants (2002):* ***TnT*** *(**statistical**)*

Brill 1992: ***Transformation-based Part-of-Speech Tagging***

Transformation-Based Tagging

Machine Learning Method (Brill 1995):

- requires annotated corpus
- automatically induces “rules” from corpus

Phase 1: tags words with general rule

Phase 2: more specific rule to correct tagging errors

Phase 3: more specific rule to correct tagging errors

Phase 4: …

Transformation-Based Tagging

Example:

The	**horse**	**will**	**win**	**the**	**race**	**tomorrow**
DT	NN	MD	VB	DT		RB

The	**horse**	**will**	**race**	**tomorrow**
DT	NN	MD		RB

Phase 1: general rule (unigram probabilistic):

race occurs more frequently as NN then as VB

Transformation-Based Tagging

Example:

The	horse	will	win	the	race	tomorrow
DT	NN	MD	VB	DT	NN	RB

The	horse	will	race	tomorrow
DT	NN	MD		RB

Transformation-Based Tagging

Example:

The	**horse**	**will**	**win**	**the**	**race**	**tomorrow**
DT	NN	MD	VB	DT	NN	RB

The	**horse**	**will**	**race**	**tomorrow**
DT	NN	MD	~~NN~~ VB	RB

Phase 2: !Transformation rule!

Change NN in VB if previous tag is MD

NN VB PREVTAG MD

Transformation-Based Tagging

How are these transformation rules learned?

1. Annotate a gold-standard corpus with general probabilistic rule
2. Investigate every possible transformation on the basis of pre-defined templates:
3.

w-3	*w-2*	*w-1*	***w***	*w1*	*w2*
t-3	t-2	t-1	**a**	t1	t2

change **a** into **b** if:

- t-1 is tag z — a b PREVTAG z
- t1 is tag z — a b NEXTTAG z
- t-2 is tag z
- t-1 or t-2 is tag z
- t-1 or t-2 or t-3 is tag z
- t-1 is tag z and t1 is tag y
- t-1 is tag z and t2 is tag y

Morphological Clues

- So far: mostly disambiguation through context
- Tagging unknown words ("*instagrammed*"): no tag candidates
- Use "morphological" clues for disambiguation (e.g. Regular expression tagger)

John : capital indicates tag NNP

computers: s-suffix indicates tag NNS

894.004.111: digits indicate tag CD

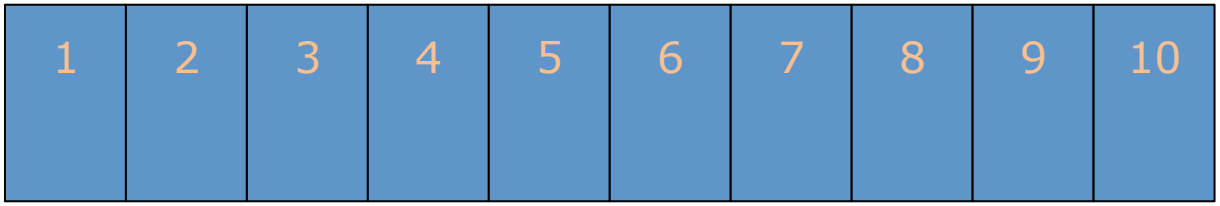
instagrammed: -ed indicates VBD/VBN

Tagging Direction

- When writing/inducing disambiguation rules, it matters whether we tag from left to right or from right to left
  - LR tagger → left context can be considered disambiguated
  - RL tagger → right context can be considered disambiguated
- We can still use non-disambiguated context in our rules, e.g. I can make this
  - If t+1 is *possibly* VB ➔ prefer MD tag

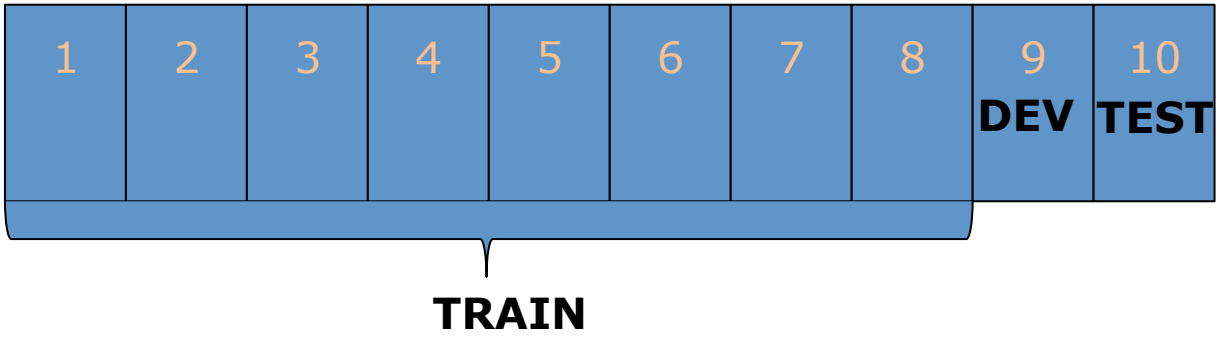
Error Analysis

- When you build a part-of-speech tagger, you want to be able to reliably estimate its accuracy
- Evaluate your system against a gold standard
  - Manually annotated text
  - Data-Driven: 10-fold cross validation



Error Analysis

- When you build a part-of-speech tagger, you want to be able to reliably estimate its accuracy
- Evaluate your system against a gold standard
  - Manually annotated text
  - Data-Driven: 10-fold cross validation



Error Analysis

- When you build a part-of-speech tagger, you want to be able to reliably estimate its accuracy
- Evaluate your system against a gold standard
  - Manually annotated text
  - Data-Driven: 10-fold cross validation
  - Compare output of tagger to gold standard annotation

Error Analysis

- **Accuracy:** % of correctly predicted pos-tags
- **Error Analysis (confusion matrix):**

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	-	.2			.7		
JJ	.2	-	**3.3**	2.1	1.7	.2	**2.7**
NN		**8.7**	-				.2
NNP	.2	**3.3**	**4.1**	-	.2		
RB	**2.2**	2.0	.5		-		
VBD		.3	.5			-	**4.4**
VBN		**2.8**				**2.6**	-

in English: weak morphology ➔ ambiguity on surface forms

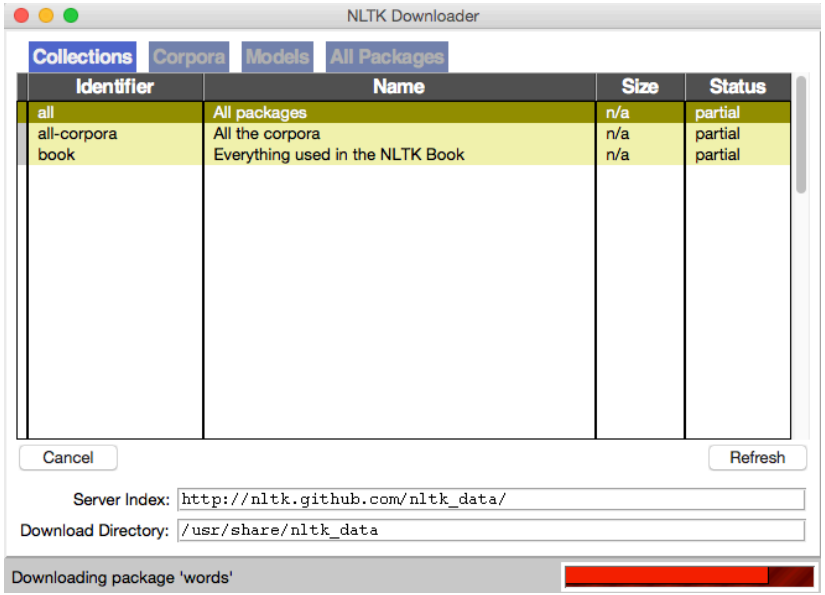
Installing NLTK

Follow the instructions for your OS on:

- http://www.nltk.org/install.html

And then download all data packages. Instructions here:

- http://www.nltk.org/data.html



```
>>> from nltk import pos_tag, word_tokenize
>>> sentence = "John's big idea isn't all that bad."
>>> tokenized = word_tokenize(sentence)
>>> tokenized
['John', "'s", 'big', 'idea', 'is', "n't", 'all', 'that', 'bad', '.']
>>> pos_tag(tokenized)
[('John', 'NNP'), ("'s", 'POS'), ('big', 'JJ'), ('idea', 'NN'), ('is', 'VBZ'),
("n't", 'RB'), ('all', 'DT'), ('that', 'DT'), ('bad', 'JJ'), ('.', '.')]
```

Try to see what the tagger can(‘t) do:

- It turns out that flies like ants.
- When the plane turns around, it flies dangerously close to the control tower.
- He made her duck for cover.

“default tagger”

```
>>> import nltk
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories='news')
>>> brown_sents = brown.sents(categories='news')
>>> brown_sents[10]
['It', 'urged', 'that', 'the', 'city', '``', 'take', 'steps', 'to', 'remedy', "''", 'this', 'problem', '.']
>>> brown_tagged_sents[10]
[('It', 'PPS'), ('urged', 'VBD'), ('that', 'CS'), ('the', 'AT'), ('city', 'NN'), ('``', '``'), ('take', 'VB'), ('steps', 'NNS'), ('to', 'TO'), ('remedy', 'VB'), ("''", "''"), ('this', 'DT'), ('problem', 'NN'), ('.', '.')]
>>> default_tagger = nltk.DefaultTagger('NN')
>>> default_tagger.tag(brown_sents[10])
[('It', 'NN'), ('urged', 'NN'), ('that', 'NN'), ('the', 'NN'), ('city', 'NN'), ('``', 'NN'), ('take', 'NN'), ('steps', 'NN'), ('to', 'NN'), ('remedy', 'NN'), ("''", 'NN'), ('this', 'NN'), ('problem', 'NN'), ('.', 'NN')]
>>> default_tagger.evaluate(brown_tagged_sents)
0.13089484257215028
```

“regex tagger”

```
>>> patterns = [
...     (r'.*ing$', 'VBG'),               # gerunds
...     (r'.*ed$', 'VBD'),                # simple past
...     (r'.*es$', 'VBZ'),                # 3rd singular present
...     (r'.*ould$', 'MD'),               # modals
...     (r'.*\'s$', 'NN$'),               # possessive nouns
...     (r'.*s$', 'NNS'),                 # plural nouns
...     (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),  # cardinal numbers
...     (r'.*', 'NN')                     # nouns (default)
... ]
>>> regexp_tagger = nltk.RegexpTagger(patterns)
>>> regexp_tagger.tag(brown_sents[10])
[('It', 'NN'), ('urged', 'VBD'), ('that', 'NN'), ('the', 'NN'), ('city', 'NN'), ('``', 'NN'),
('take', 'NN'), ('steps', 'NNS'), ('to', 'NN'), ('remedy', 'NN'), ("''", 'NN'), ('this',
'NNS'), ('problem', 'NN'), ('.', 'NN')]
>>> regexp_tagger.evaluate(brown_tagged_sents)
0.2032639178948624 5
```

“unigram tagger”

```
>>> unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
>>> unigram_tagger.tag(brown_sents[10])
[('It', 'PPS'), ('urged', 'VBD'), ('that', 'CS'), ('the', 'AT'), ('city', 'NN'), ('``', '``'),
('take', 'VB'), ('steps', 'NNS'), ('to', 'TO'), ('remedy', 'VB'), ("''", "''"), ('this',
'DT'), ('problem', 'NN'), ('.', '.')]
>>> unigram_tagger.evaluate(brown_tagged_sents)
0.934900650396801 7
```

Warning: a huge methodological mistake was made in the experiment above

!!!!!!!EVALUATING on TRAINING SET IS FORBIDDEN!!!!!

```
>>> brown_train = list(brown.tagged_sents(categories='news')[:-500])
>>> brown_test = list(brown.tagged_sents(categories='news')[-500:])
>>> unigram_tagger = nltk.UnigramTagger(brown_train)
>>> unigram_tagger.evaluate(brown_test)
0.810496165573316
```

We take the last 500 sentences (rougly 10%) as the test set

"brill tagger"

- Install PyYAML: http://pyyaml.org/wiki/PyYAML
- Download brill2.py from http://www.clips.uantwerpen.be/cl1415 and place it in
  C:\Python3.*\Lib\site-packages\nltk\tag (Windows)
  /Library/Frameworks/Python.framework/Versions/3.*/lib/python3.*/site-packages/nltk-3.*/nltk/tag (MAC)
  ***: depends on your local installation**

```
>>> from nltk.corpus import brown
>>> from nltk.tag import UnigramTagger
>>> from nltk.tag.brill2 import SymmetricProximateTokensTemplate, ProximateTokensTemplate
>>> from nltk.tag.brill2 import ProximateTagsRule, ProximateWordsRule, FastBrillTaggerTrainer
>>> brown_train = list(brown.tagged_sents(categories='news')[:-500])
>>> brown_test = list(brown.tagged_sents(categories='news')[-500:])
>>> unigram_tagger = UnigramTagger(brown_train)
```

“brill tagger”

```
>>> templates = [
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (1,1)),     # t-1 or t1 is tag z
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (2,2)),     # t-2 or t2 is tag z
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (1,2)),     # t-1 or t1 or t-2 or t-2 is tag z
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (1,3)),     # t-1 or t1 or t-3 or t3 is tag z
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (1,1)),    # w-1 or w1 is word w
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (2,2)),    # w-2 or w2 is word w
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (1,2)),    # w-1 or w1 or w-2 or w2 is w
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (1,3)),    # w-1 or w1 or w-3 or w3 is w

...     ProximateTokensTemplate(ProximateTagsRule, (-1, -1), (1,1)),  # t1 and t-1 are tag y and z
...     ProximateTokensTemplate(ProximateWordsRule, (-1, -1), (1,1)), # w1 and w-1 are word v and w
... ]
>>> trainer = FastBrillTaggerTrainer(initial_tagger=unigram_tagger,templates=templates,
trace=3,deterministic=True)
>>> brill_tagger = trainer.train(brown_train)
<…>
```

“brill tagger”

```
>>> brill_tagger.tag(brown_train[10])
[(('It', 'PPS'), None), (('urged', 'VBD'), None), (('that', 'CS'), None), (('the', 'AT'), None),
(('city', 'NN'), None), (('``', '``'), None), (('take', 'VB'), None), (('steps', 'NNS'), None),
(('to', 'TO'), None), (('remedy', 'VB'), None), (("''", "''"), None), (('this', 'DT'), None),
(('problem', 'NN'), None), (('.', '.'), None)]

>>> brill_tagger.evaluate(brown_test)
0.8347962672087221
```

vs.

```
>>> unigram_tagger.evaluate(brown_test)
0.810496165573316
```

pickle

With pickle, you can “save” and “load” objects, saving you a lot of time.

```
>>> import nltk
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories='news')
>>> tagger = nltk.UnigramTagger(brown_tagged_sents)
>>> import pickle
>>> pickle.dump( tagger, open( "unigramtagger.p", "wb" ) )
```

New session

```
>>> import pickle
>>> tagger = pickle.load(open( "unigramtagger.p", "rb" ))
>>> tagger.tag(['flies','like','an','arrow'])
[('flies', 'VBZ'), ('like', 'CS'), ('an', 'AT'), ('arrow', None)]
```

A tagger for Dutch?

NLTK also has Dutch corpora, e.g. alpino

```
>>> import nltk
>>> from nltk.corpus import alpino
>>> alpino_tagged_sents = alpino.tagged_sents()
>>> len(alpino_tagged_sents)
7136
>>> alpino_tagged_sents[5]
[('mag', 'verb'), ('de', 'det'), ('Bondsrepubliek', 'noun'), ('In', 'prep'), ('plaats', 'prep'), ('van', 'prep'), ('het', 'det'), ('stelsel', 'noun'), ('van', 'prep'), ('importheffingen', 'noun'), ('en', 'vg'), ('exportsubsidies', 'noun'), ('wel', 'adv'), ('de', 'det'), ('grenzen', 'noun'), ('sluiten', 'verb'), ('voor', 'prep'), ('granen', 'noun'), ('en', 'vg'), ('zuivelprodukten', 'noun'), ('.', 'punct')]
```

Figure out how to build a unigramtagger and a brill tagger for Dutch. Perform a methodologically correct evaluation (1 fold)

```
>>> alpino_train = alpino_tagged_sents[0:6500]
>>> alpino_test = alpino_tagged_sents[6500:]
>>> len(alpino_train)
6500
>>> len(alpino_test)
636
```

A tagger for Dutch?

```
>>> alpino_train = alpino_tagged_sents[0:6500]
>>> alpino_test = alpino_tagged_sents[6500:]
>>> len(alpino_train)
6500
>>> len(alpino_test)
636
>>> dutchunigram = nltk.UnigramTagger(alpino_train)
>>> dutchunigram.evaluate(alpino_test)
0.8090113243916151 6
```

A tagger for Dutch?

```
>>> from nltk.tag.brill2 import SymmetricProximateTokensTemplate, ProximateTokensTemplate
>>> from nltk.tag.brill2 import ProximateTagsRule, ProximateWordsRule, FastBrillTaggerTrainer
>>> templates = [
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (1,1)),
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (2,2)),
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (1,2)),
...     SymmetricProximateTokensTemplate(ProximateTagsRule, (1,3)),
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (1,1)),
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (2,2)),
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (1,2)),
...     SymmetricProximateTokensTemplate(ProximateWordsRule, (1,3)),
...     ProximateTokensTemplate(ProximateTagsRule, (-1, -1), (1,1)),
...     ProximateTokensTemplate(ProximateWordsRule, (-1, -1), (1,1)),
... ]
>>> trainer = FastBrillTaggerTrainer(initial_tagger=dutchunigram,templates=templates,
trace=3,deterministic=True)
>>> brill_tagger = trainer.train(alpino_train)
Training Brill tagger on 6500 sentences...
<…>
>>> brill_tagger.evaluate(alpino_test)
0.8226648461970926
```

Assignment

Download www.clips.ua.ac.be/cl1415/participles.py

Part-of-Speech Tagging

1. Train a unigram or brill part-of-speech tagger for English and write a function that tokenizes the text and returns a list of the verbs in the text.

Extra credit: on http://www.nltk.org/book_1ed/ch05.html, section "Combining Taggers", the developers of nltk explain how you can enhance tagging performance by using "backoff" taggers. Try and see if you can build such a combined tagging system.

DEADLINE: 8 December 2014

Send python code through e-mail to guy.depauw@uantwerpen.be
Don't hesitate to contact your helpline guy.depauw@uantwerpen.be